Comparaison intensive des séquences protéiques : indexation des banques et usage des instructions SIMD des microprocesseurs

van Hoa Nguyen, Dominique Lavenier

Comparaison intensive des séquences protéiques : indexation des banques et usage des instructions SIMD des microprocesseurs

Van Hoa Nguyen, Dominique Lavenier

IRISA, Campus de Beaulieu, Rennes, France
{vhnguyen, lavenier}@irisa.fr

Abstract: *This paper presents a method for speeding up the comparison of two protein banks. It is based on two techniques: bank indexing and fast computation of short alignments using the SIMD instruction set of modern microprocessors. The algorithm follows the seed-based approach in which hits are first exhibited before further processing. Performances are compared with BLASTP and show a speed-up of 4 with comparable sensitivity.*

1 Introduction

La recherche de similarité entre séquences génomiques est une des tâches fondamentales de la bioinformatique. Depuis 25 ans, de nombreux efforts ont été entrepris pour réduire la complexité de l'algorithme de Smith et Waterman qui devient hors de portée dès lors que des volumes de données importants sont en jeu. Parmi toutes les innovations, l'heuristique implémentée dans la famille de programmes BLAST, reste celle qui permet une réduction drastique des temps d'exécution tout en procurant une sensibilité satisfaisante pour de nombreuses applications. L'heuristique se base sur le fait qu'une grande majorité des alignements possèdent des régions de forte similarité où l'on peut mettre en évidence, dans les deux portions de séquences qui le composent, des mots identiques de W caractères. Il s'agit alors de repérer ces mots (appelés graines) qui constituent des points d'ancrage à partir desquels on peut calculer un alignement plus conséquent.

La parallélisation fine de ce type d'algorithme reste très difficile. En général, pour accélérer les traitements impliquant ces algorithmes de comparaison, une parallélisation à gros grain est effectuée : les données sont dispatchées sur plusieurs nœuds d'une machine parallèle, chaque nœud travaillant indépendamment des autres avant de fusionner leurs résultats.

Dans ce papier, nous introduisons une parallélisation fine basée sur l'usage des instructions SIMD des processeurs actuels. Plutôt que d'effectuer une opération arithmétique et logique sur 128 bits en un seul cycle machine, ces processeurs offrent la possibilité de fragmenter un opérande de 128 bits en 8 opérandes de 16 bits en 16 opérandes de 8 bits, et de réaliser simultanément 8 ou 16 opérations. La contrainte, bien sûr, est que les valeurs manipulées restent dans une dynamique réduite qui ne dépasse pas 8 ou 16 bits.

Ce parallélisme à grain fin peut avantageusement être exploité lors des premières phases du calcul des alignements où, à partir des points d'ancrage, un score est calculé par rapport au voisinage immédiat pour décider s'il faut poursuivre le calcul. La valeur du score est suffisamment faible pour être encapsulée dans un entier codé sur 8 bits. Le problème est alors d'avoir simultanément un grand nombre de scores à calculer. L'indexation des banques nous apporte la solution. Pour une graine donnée, toutes ses positions sont répertoriées dans les deux banques. Ainsi, si `n1` et `n2` sont respectivement le nombre d'occurrences d'une graine dans la banque 1 et dans la banque 2, on doit effectuer `n1 x n2` extensions. Cet ensemble de calculs peut alors être efficacement implémenté via les instructions SIMD du microprocesseur.

2 Structure du programme

Le programme baptisé iBLASTP se décompose en 5 étapes :

Etape 1 : indexation. L'indexation est réalisée sur les 2 banques. Elle consiste à stocker les occurrences des positions des 20^W graines possibles dans les banques si W représente leur taille (en nombre d'acides aminés).

Etape 2 : construction de listes de voisinage. Le programme considère chaque graine possible. Pour chacune d'elle, il construit 2 listes de sous séquences correspondant au voisinage immédiat de toutes ses occurrences dans les 2 banques.

Etape 3 : extension sans gap. Pour une graine donnée, l'étape précédente à produit 2 listes de sous séquences de n1 et n2 éléments. L'extension consiste à calculer un score par rapport à la similarité des voisinages. Si le score dépasse un seuil S_1, ce début d'alignement est passé à l'étape 4.

Etape 4 : extension avec gap. Cette étape est en fait divisée en 2 sous étapes (4.1 et 4.2), la première intervenant comme filtre pour la seconde. Dans un premier temps, l'idée est de limiter l'espace de recherche de la programmation dynamique. Cette procédure est donc contrainte à la fois sur le nombre d'erreurs de gaps autorisés et sur la taille de l'alignement. Si le score de l'alignement ainsi calculé dépasse un seuil S_2, alors la procédure de programmation dynamique standard est lancée (sous étape 4.2).

Etape 5 : affichage des alignements. Cette dernière étape met en forme les alignements et affiche les résultats.

Dans ce programme, les instructions SIMD sont utilisées dans deux endroits qui concentrent plus de 80 % du temps de calcul : l'étape 3 et l'étape 4.1. L'extension sans gap calcule 16 scores en parallèle tandis que l'extension contrainte avec gap ne peut qu'effectuer que 8 calculs de scores en parallèle, la dynamique pouvant dépasser 8 bits. Il faut noter que, dans les deux cas, les calculs se font sur des tailles de sous séquences fixes.

3 Résultats

Le tableau ci-dessous compare les temps d'exécution et la sensibilité par rapport au programme BLASTP du NCBI (paramètres par défaut, e-value = 10^{-3}). La banque 1 contient 141708 séquences issues de PIR. La taille de la banque 2 varie de 5000 à 40000 séquences extraites de SwissProt. Les tests ont été effectués sur un processeur Intel à 2,6 GHz, 2 Go de RAM, fonctionnant sous Linux.

Nb. Séq. (SwissProt)	BLASTP (sec)	iBLASTP (sec)	accélération	BLASTP (nb. align.)	iBLASTP (nb. align.)	% align. identique
10.000	6.832	1.585	4,3	611.031	611.093	95,3 %
20.000	13.597	3.188	4,2	1.047.794	1.059.822	95,0 %
40.000	26.111	6.053	4,3	2.237.076	2.237.422	95,2 %

Table 1. Temps d'exécution et nombre d'alignements de BLASTP et de iBLASTP.

En moyenne, nous mesurons un facteur d'accélération supérieur à 4. Le nombre d'alignements produits est comparable à celui de BLASTP. Environ 95% des alignements sont identiques, les 5 % restant émanant des heuristiques sensiblement différentes mises en œuvre dans les 2 cas.

Les performances obtenues sont dues à un usage optimisé de l'architecture interne des microprocesseurs via les instructions SIMD. Leurs mises en œuvre ne font pas appel à des outils particuliers : un compilateur standard suffit. Cependant, leur efficacité dépend de la capacité à extraire un parallélisme à grain fin dans le code des programmes. Dans le cas présent, cette possibilité passe par un remaniement complet du code des algorithmes de comparaison en indexant toutes les données en mémoire.